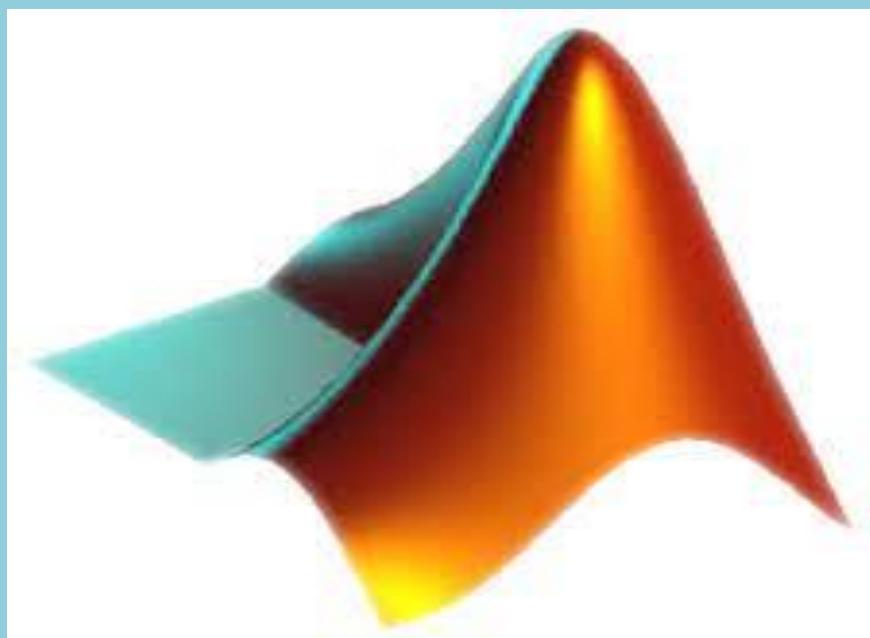


# Initiation à Matlab

Ali El Mahraoui

Faculté des sciences et techniques à Al-Hoceima

MIP-S4



## Table des matières

1	Introduction.....	3
1.1	Définition.....	3
1.2	Domaines d'utilisation.....	3
1.3	Modes de fonctionnement.....	3
1.4	Présentation de l'interface .....	4
2	Les commandes de base.....	5
2.1	Les opérations de base .....	5
2.2	Les nombres réels.....	7
2.3	Les nombres complexes .....	7
2.4	Formats d'affichage.....	8
2.5	Dialogue E/S via clavier .....	8
3	Les variables .....	10
3.1	Déclaration d'une variable .....	10
3.2	Constantes prédéfinies dans Matlab.....	11
3.3	Les types de variables.....	11
3.4	Espace de travail.....	13
3.5	Les vecteurs .....	14
3.5.1	Vecteur ligne.....	14
3.5.2	Vecteur colonne .....	14
3.5.3	Concaténation des vecteurs .....	15
3.5.4	Manipulation les éléments d'un vecteur.....	15
3.5.5	Manipulation des éléments d'un vecteur .....	<b>Erreur ! Signet non défini.</b>
3.5.6	Vecteurs spéciaux.....	16
3.5.7	Opérations sur les vecteurs.....	18
3.6	Les matrices.....	19
3.6.1	Définition .....	20
3.6.2	Les matrices spéciales .....	20
3.6.3	Extraction des éléments d'une matrice.....	22
3.6.4	Opérations et fonctions appliquées sur les matrices .....	26
4	Représentations graphiques Graphique 2D et 3D.....	28

# 1 Introduction

## 1.1 Définition

- ✓ Le nom MATLAB est la contraction du terme anglais **Matrix Laboratory** a été développé par la société Mathworks pour faire du calcul matriciel.
- ✓ MATLAB devient un outil puissant pour la programmation et un environnement de développement.
- ✓ MATLAB est un langage de programmation qui a l'avantage :
  - D'être simple à utiliser : ne nécessite pas une compilation (langage interpréteur ; les instructions sont interprétées et exécutées ligne par ligne) et les variables utilisées sont déclarées implicitement.
  - De contenir une riche bibliothèque de fonction dans divers domaines (analyse numérique, statistique, représentation graphique ...).
  - D'avoir une interface graphique maniable.

## 1.2 Domaines d'utilisation

- ✓ L'algèbre matricielle : résolutions numériques des systèmes linéaires et non linéaires, le calcul numérique des valeurs et des vecteurs propres...
- ✓ La résolution numérique des EDP.
- ✓ le traitement du signal et de l'image.
- ✓ ...

## 1.3 Modes de fonctionnement

Il existe deux modes de fonctionnement :

1. **Mode interactif** : MATLAB exécute les instructions au fur et à mesure qu'elles sont données par l'utilisateur.
2. **Mode exécutif** : MATLAB exécute ligne par ligne un "fichier .m"

## 1.4 Présentation de l'interface



Chaque zone possède un objectif précis :

- Le menu regroupe des commandes de base de Matlab, comme enregistrer, afficher les préférences, etc...
- L'explorateur de fichiers permet de visualiser ses fichiers scripts et de les ouvrir pour les éditer.
- La zone de commande permet d'écrire des commandes et de visualiser leur résultat
- La zone des variables permet de visualiser toutes les variables en mémoire à l'instant présent (leur nom ainsi que visualiser leur contenu).
- L'historique permet de visualiser l'historique des commandes précédemment exécutées.

## 2 Les commandes de base

### 2.1 Les opérations de base

➤ Matlab comme calculette :

```
>>6+7
```

```
ans =
```

```
13
```

```
>>
```

ans est une variable qui contient toujours le résultat de la dernière opération.

MATLAB utilise les opérateurs usuels suivants pour manipuler les expressions :

Opérateur	Définition
+	Addition
-	Soustraction
*	Multiplication
^	Puissance
/	Division
\	Division à gauche
'	Transposé

Il utilise aussi des opérateurs logiques et relationnels :

Opérateurs	Définition
<= >=	inférieur (supérieur) ou égal

<code>==</code>	Egal
<code>~=</code>	différent
<code>&amp;</code>	et logique
<code> </code>	ou logique
<code>~</code>	complément logique(not)
<code>Xor</code>	ou exclusif

Voici quelques fonctions mathématiques et leurs symboles

Symboles	Description
<code>exp(x)</code>	exponentielle de x
<code>log(x)</code>	logarithme népérien de x
<code>log10(x)</code>	logarithme en base 10 de x
<code>x^n</code>	x à la puissance n
<code>sqrt(x)</code>	racine carrée de x
<code>abs(x)</code>	valeur absolue de x
<code>sign(x)</code>	1 si $x > 0$ et 0 si $x \leq 0$
<code>sin(x)</code>	sinus de x
<code>cos(x)</code>	cosinus de x
<code>tan(x)</code>	tangente de x
<code>asin(x)</code>	sinus inverse de x (arc sinus de x)
<code>sinh(x)</code>	sinus hyperbolique de x
<code>asinh(x)</code>	sinus hyperbolique inverse de x

## 2.2 Les nombres réels

Comme dans tous les langages de programmation, les nombres réels s'écrivent avec un point pour séparer la partie entière de la partie décimale.

```
>> 2.7*3.9
```

ans=

10.5300

## 2.3 Les nombres complexes

\* Les nombres complexes peuvent être écrits sous forme cartésienne ou polaire :

✓ Forme cartésienne :

$1+3i$  ;  $0.3+2*i$  ;  $-2+0.4*j$ ;  $6+j*0.8$

✓ Forme polaire :

$2.35*\exp(0.25*j)$

Avec  $i^2 = j^2 = -1$

Les fonctions appliquées aux nombres complexes :

Fonctions	Définitions
$\text{conj}(X)$	conjugué du nombre complexe X
$\text{real}(X)$	partie réelle
$\text{imag}(X)$	partie imaginaire
$\text{abs}(X)$	module
$\text{angle}(X)$	argument (en radians)

## 2.4 Formats d'affichage

Format	Affichage jusqu'à
<b>Short</b>	Affichage jusqu'à 4 chiffres après la virgule
<b>Long</b>	Affichage jusqu'à 16 chiffres après la virgule
<b>short e</b>	Affichage jusqu'à 5 chiffres plus l'exposant
<b>long e</b>	Affichage jusqu'à 16 chiffres plus l'exposant
<b>Bank</b>	Affichage jusqu'à 2 chiffres après la virgule
<b>Rat</b>	Affichage rationnel
<b>Hex</b>	Affichage dans la base 16.

Exemple :

<pre>&gt;&gt; format short &gt;&gt; pi ans =       3.1416 &gt;&gt; format long &gt;&gt; 5/9 ans =       0.5555555555555556 &gt;&gt; format rat &gt;&gt; 0.62 ans =       31/50</pre>	<pre>&gt;&gt; format short e &gt;&gt; 5/9 ans =       5.5556e-01 &gt;&gt; format long e &gt;&gt; 5/9 ans =       5.555555555555556e-01 &gt;&gt; format hex &gt;&gt; 5/9 ans =       3fe1c71c71c71c72 &gt;&gt; format bank &gt;&gt; 5/9 ans =       0.56</pre>
--	---

## 2.5 Dialogue E/S via clavier

Pour instruire au programme d'attendre la saisie d'une valeur pour 'a' on utilise l'instruction **input** :

```
>> a=input('La valeur de a est :')
```

la valeur de a est : 54

a =

54

```
>> Nom = input('saisir votre nom :')
```

saisir votre nom : 'ali'

Nom =

ali

```
>> Nom = input('saisir votre nom :','s')
```

saisir votre nom : ali

Nom =

Ali

Pour afficher les données, on écrit directement le nom de la variable, ou bien on utilise les fonctions disp() et fprintf() :

```
>> disp('la valeur de a est :'), disp(a)
```

la valeur de a est :

54

```
>> a=5/9;
```

```
>> fprintf('la valeur de a est %.5f \n',a)
```

La valeur de a est 0.55556

```
>> A = 20 ;
```

```
>> B = 15 ;
```

```
>> fprintf('%d est plus petit que %d\n', B, A)
```

15 est plus petit que 20

### 3 Les variables

#### 3.1 Déclaration d'une variable

Pour déclarer une variable il suffit d'utiliser l'opérateur =

```
>>x=5
```

x =

5

- Cette commande crée une variable x et lui affecte la valeur 5.
- Cette valeur est ajoutée au workspace.

En général :

- Toutes les variables sont des matrices. L'indexation est (ligne \* colonne).
- Pour séparer les lignes, on met un point-virgule ou un retour à la ligne en tapant entrer.
- Pour séparer les colonnes, on met une virgule ou espace blanc.

#### Exemples :

X=3                variable scalaire (1 \*1)

Y = 2.3549        variable scalaire (1 \*1)

V= [1,2,3,4]      vecteur ligne (1\*4)

U= [2;6;8]        vecteur colonne (3\*1)

S='une chaîne de caractères' vecteur ligne (1\*24)

A= [1 2;3 4]      matrice de taille (2\*2)

- ✓ Dans Matlab la déclaration de type des variables est facultatif.
- ✓ Les types sont déterminés lorsque les variables sont initialisées.
- ✓ Toute variable numérique est à virgule flottante de double précision et les textes sont des chaînes de caractères.

- ✓ Matlab est sensible à la casse (a # A).

### 3.2 Constantes prédéfinies dans Matlab

La constante	sa valeur
pi	3.1415...
inf	$\infty$
NaN	Not a Number
eps	2.2204 e-16
Exp(1)	2.7183....
realmin	2.225 e-308
realmax	1.7977 e +308

### 3.3 Les types de variables

Types de données	Taille (bytes)	Gammes de valeurs	Fonctions de conversion
<i>Signed 8-bit integer</i>	1	$-2^7$ à $2^7 - 1$	int8()
<i>Signed 16-bit integer</i>	2	$-2^{15}$ à $2^{15} - 1$	int16()
<i>Signed 32-bit integer</i>	4	$-2^{31}$ à $2^{31} - 1$	int32()
<i>Signed 64-bit integer</i>	8	$-2^{63}$ à $2^{63} - 1$	int64()
<i>Unsigned 8-bit integer</i>	1	0 à $2^8 - 1$	uint8()
<i>Unsigned 16-bit integer</i>	2	0 à $2^{16} - 1$	uint16()
<i>Unsigned 32-bit integer</i>	4	0 à $2^{32} - 1$	uint32()

<i>Unsigned 64-bit integer</i>	8	0 à $2^{64}-1$	uint64()
Double-Precision Floating Point (64 bits)	8	$[-1.7977e+308, -2.22507e-308]$ $[2.22507e-308, 1.7977e+308]$	double()
Single-Precision Floating Point (32 bits)	4	$-3.40282 \times 10^{38}$ à $-1.1755 \times 10^{-38}$ $1.1755 \times 10^{-38}$ à $3.40282 \times 10^{38}$	single()
Char	2	0 à 65535	char()
Logical (8 bits)	1	0 à 1	logical()

Exemples :

```
>> x=14 ;
>> y=1.2357 ;
>>z=int16(x) ;
>>t='السلام عليكم' ;
>>whos
```

Name	Size	Bytes	Class	Attributes
z	1*1	2	int16	
x	1*1	8	double	
y	1*1	8	double	
t	1*12	24	char	

Fonction	Rôle
<b>mat2str</b>	Convertit une matrice en chaîne de caractère
<b>num2str</b>	Convertit un nombre en une chaîne de caractères
<b>str2double</b>	Convertit une chaîne en nombre double précision
<b>str2num</b>	Convertit une chaîne de caractères en nombre
<b>Bin2dec</b>	Convertit un nombre binaire en décimal

<b>dec2bin</b>	Convertit un nombre décimal en binaire
<b>dec2hex</b>	Convertit un nombre décimal en hexadécimal
<b>hex2dec</b>	Convertit un nombre hexadécimal en décimal
<b>hex2num</b>	Convertit un nombre sous forme de chaîne de caractères hexadécimal en un nombre décimal double précision
<b>num2hex</b>	Convertit des nombres simples et double précision en chaîne de caractères hexadécimale

Exemples :

<b>1)</b> <pre>&gt;&gt; nom = 'Houda' ; age = 80 ; &gt;&gt; P = [nom, 'aura', num2str(age), 'ans', 'cette année'] ; &gt;&gt; disp(P) %Affichera Houda aura 80 ans cette année</pre>	<b>2)</b> <pre>&gt;&gt; hex2dec('2B9') %Donnera : ans = 697</pre>
--	--

### 3.4 Espace de travail

Les variables sont stockées dans l'espace de travail (workspace) et peuvent être utilisées dans les calcul subséquent.

Commande de workspace	rôle
Who	Affiche les noms des variables actives
Whos	Affiche les noms des variables actives avec d'autres informations(taille, type)
clc	Nettoie la fenêtre des commandes

clear	Nettoie l'espace de travail (supprime toutes les variables)
clear a b c	Supprime les variables a, b et c
save nom-fic	Enregistre toutes les variables de l'espace de travail dans le fichier nom-fic.mat
load nom-fic	Ramener dans l'espace de travail les variables sauvegardées dans le fichier nom-fic.mat
Help <b>commande</b>	pour une explication de l'utilisation de la commande Matlab <b>commande</b>
lookfor <b>sujet</b>	pour une liste des commandes Matlab qui concernent le sujet <b>sujet</b>

## 3.5 Les vecteurs

### 3.5.1 Vecteur ligne

- ✓ On définit un vecteur ligne en donnant la liste de ses éléments entre crochet [ ].

```
>> x=[1,5,8,9]           ou >> x=[1 5 8 9]
x=
    1   5   8   9
>> length(x)
ans =
    4
```

### 3.5.2 Vecteur colonne

- ✓ On définit un vecteur colonne en donnant la liste de ses éléments entre crochet [ ], en les séparant par point-virgule ou bien en cliquant sur entré,

```
>> x=[1;5;8;9]           ou >> x=[1
x=                           5
```

```

1                               8
5                               9]
8                               x=
9                               1
>> length(x)                  5
ans =                         8
4                               9
>> y=x'                      >> y=x'
y=                           1   5   8
                                9

```

### 3.5.3 Concaténation des vecteurs

```

>> v1=[1 2 3]; v2=[ 4 5 6]; v3=[7 8 9];
>> v=[v1 v2 v3]
v=
1   2   3   4   5   6   7   8   9
>> length(v)
ans =
9

```

On peut créer le vecteur v de cette façon

```

>> v = linspace(1, 9, 9)
v =
1   2   3   4   5   6   7   8   9

```

### 3.5.4 Manipulation les éléments d'un vecteur

On peut utiliser les éléments d'un vecteur grâce à leurs indices dans le tableau.

Le k<sup>ème</sup> élément d'un vecteur est désigné par x(k).

L'indice de premier élément est obligatoirement 1.

```

>> v(5)
ans =
5

```

```
>>v(0)
```

Subscript indices must either be real positive integers or logicals

```
>>v(1)
```

```
ans =
```

```
1
```

- ✓ Il est possible de manipuler plusieurs éléments d'un vecteur en même temps. Les éléments de  $k$  à  $l$  du vecteur  $x$  sont représentés par  $x(k : l)$ .
- ✓ Si on veut extraire les éléments dont les indices sont en progression arithmétique  $k, k+p, k+2p, \dots, k+np=l$ , on écrit  $x(k:p:l)$ .
- ✓ En générale, si  $k$  est un vecteur de valeurs entières, pour extraire les éléments du vecteur  $x$  dont les indices sont les élément de vecteur  $k$ , on écrit  $x(k)$ .

#### Exemple :

```
>>v=[2 4 6 8 10]
```

```
v =
```

```
2     4     6     8     10
```

```
>> v(3)    % 3ème élément
```

```
ans =
```

```
6
```

```
>> v(2:4)   % sous-vecteur du 2ème au 4ème élément
```

```
ans =
```

```
4     6     8
```

```
>> v(2:end)  %sous-vecteur du 2ème au dernier élément
```

```
ans =
```

```
4     6     8     10
```

```
>> v([5 3 1])  % les éléments v(5), v(3), et v(1)
```

```
ans =
```

```
10    6     2
```

### 3.5.5 Vecteurs spéciaux

`ones (1, n)` : Vecteur ligne de longueur  $n$  dont tous les éléments valent 1

`ones (m, 1)` : Vecteur colonne de longueur  $m$  dont tous les éléments valent 1

<code>zeros(1, n)</code>	:	Vecteur ligne de longueur n dont tous les éléments valent 0
<code>zeros(m,1)</code>	:	Vecteur colonne de longueur m dont tous les éléments valent 0
<code>rand(1,n)</code>	:	Vecteur ligne de longueur n dont les éléments sont générés de manière aléatoire entre 0 et 1
<code>rand(m,1)</code>	:	Vecteur colonne de longueur m dont les éléments sont générés de manière aléatoire entre 0 et 1

**Exemples :**

```
>> v = zeros(1,5)
v =
0 0 0 0 0
```

```
>> v = zeros(5,1)
v =
0
```

```
0
0
0
0
0
```

```
>> v = ones(1,5)
v =
1 1 1 1 1
```

```
>> v = ones(5,1)
v =
1
```

```
1
1
1
1
1
```

```
>> v1 = rand(1, 5) % 5 valeurs aléatoires (0-1)
```

```
v1 =
0.0975    0.2785    0.5469    0.9575    0.9649
```

```

>> v2 = rand(1, 5) % 5 autres valeurs aléatoires (0-1)
v2 =
0.1576    0.9706    0.9572    0.4854    0.8003

```

### 3.5.6 Opérations sur les vecteurs

Opération	Signification	Exemple avec $\mathbf{u} = [-2, 6, 1];$ $\mathbf{v} = [3, -1, 4];$
+	Addition des vecteurs	$\mathbf{u} + \mathbf{v}$ ans = 0 8 3
-	Soustraction des vecteurs	$\mathbf{u} - \mathbf{v}$ ans = -4 4 -1
.*	Multiplication élément par élément	$\mathbf{u} * \mathbf{v}$ ans = -6 -6 4

./	Division élément par élément	>> u/2 ans = -1.0000 3.0000 0.5
.^	Puissance élément par élément	>> u.^2 ans = 4 36 >> u.^v ans = -8.0000 0.1667 1.0000
sqrt(x)	Racine carrée de vecteur x	>>x=[1 4 9]; >>sqrt(x) Ans = 1 2 3

- ✓ **cross(u,v) : le produit vectoriel de vecteurs u et v,**
- ✓ **sum(u) : la somme des éléments de u,**
- ✓ **sum(u.\*v) : le produit scalaire de u et v,**
- ✓ **prod(u) : le produit des éléments de u,**
- ✓ **max(u) : le plus grand élément de u,**
- ✓ **min(u) : le plus petit élément de u,**
- ✓ **mean(u) : le moyen des éléments de u,**
- ✓ **sort(u) : ordonne les éléments de u par ordre croissant,**
- ✓ **fliplr(u) : renverse l'ordre des éléments de u.**

### 3.6 Les matrices

### 3.6.1 Définition

- ✓ On définit une matrice en donnant la liste de ses éléments entre crochets [ ].
- ✓ On sépare les éléments de ses lignes par virgule ou bien par espace blanc.
- ✓ On sépare les éléments de ses colonnes par point-virgule ou bien par entrer.

```
>>A=[1 2;4 5]
```

A=

1	2
4	5

### Les éléments d'une matrice

- ✓ Un élément d'une matrice est localisé par ses numéros de ligne et de colonne.  $A(i, j)$  désigne le  $j$ ème élément de la ligne  $i$  de la matrice  $A$ .
- ✓  $A(2,1)$  désigne le premier élément de la deuxième ligne de  $A$ .

```
>>A=[1 2;4 5];
```

```
>> A(2,1)
```

ans =

4

### 3.6.2 Les matrices spéciales

Matrice	Définition

A = eye(n)	A matrice identité n x n
A = zeros(n,m)	A matrice de dimension n x m dont tous les éléments valent 0
A = ones(n,m)	A matrice de dimension n x m dont tous les éléments valent 1
A = diag(v)	Si v est un vecteur : A est une matrice carrée avec les éléments du vecteur v dans la diagonale de A . Si v est une matrice : A est un vecteur extraite de la diagonale de A
A = rand(n,m)	A matrice aléatoire de dimension n x m dont les éléments sont générés aléatoirement entre 0 et 1
A= magic(n)	Qui permet d'obtenir une matrice magique d'ordre n

### Exemples :

```

>> eye(3)
ans =
  1   0   0
  0   1   0
  0   0   1
>> ones(3,2)
ans =
  1   1
  1   1
  1   1
>> zeros(2)
ans =
  0   0
  0   0
>> rand(2,3)
ans =

```

```

0.4565  0.8214  0.6154
0.0185  0.4447  0.7919
>> magic(6)
ans =
35   1   6   26  19  24
3   32  7   21  23  25
31   9   2   22  27  20
8   28  33  17  10  15
30   5   34  12  14  16
4   36  29  13  18  11

```

### 3.6.3 Extraction des éléments d'une matrice

- ✓ Pour extraire i-ème ligne de la matrice A, on utilise  $A(i, :)$ .
- ✓ Pour extraire j-ème colonne de la matrice A, on utilise  $A(:, j)$

#### Exemple :

```

>> A = magic(5)
A =
17  24   1   8   15
23   5   7   14  16
 4   6  13  20  22
10  12  19  21   3
11  18  25   2   9
>> A(1, :)
ans =
17  24   1   8   15
>> A(:, 2)
ans =
24
5
6
12
18

```

```
>> A = magic(5)
```

```
A =
```

```
17 24 1 8 15  
23 5 7 14 16  
4 6 13 20 22  
10 12 19 21 3  
11 18 25 2 9  
>> L = [1 3 5]; C = [3 4];
```

```
>> A(L,C)
```

```
ans =
```

```
1 8  
13 20  
25 2
```

```
>> A(1:2:5 , 3:4)
```

```
ans =
```

```
1 8  
13 20  
25 2
```

Si l'on souhaite échanger les colonnes 2 et 3 de la matrice A une première possibilité consiste à exécuter:

```
>> v = A(:,2); A(:,2) = A(:,3); A(:,3) = v;
```

```
A =
```

```
17 1 24 8 15  
23 7 5 14 16  
4 13 6 20 22  
10 19 12 21 3  
11 25 18 2 9
```

Il existe des commandes MATLAB permettant de manipuler globalement des matrices.

- ✓ La commande **diag** permet d'extraire la diagonale d'une matrice: si A est une matrice, **v=diag(A)** est le vecteur composé des éléments diagonaux de A.

- ✓ Elle permet aussi de créer une matrice de diagonale fixée: si  $v$  est un vecteur de dimension  $n$ ,  $A=\text{diag}(v)$  est la matrice diagonale dont la diagonale est  $v$ .

```
>> A=eye(3);
```

```
>>diag(A)
```

```
ans =
```

```
1
```

```
1
```

```
1
```

```
>> v=[1:3]
```

```
v =
```

```
1 2 3
```

```
>> diag(v)
```

```
ans =
```

```
1 0 0
```

```
0 2 0
```

```
0 0 3
```

La commande **diag** admet un second paramètre  $k$  pour désigner la  $k$  sur-diagonale (si  $k>0$ ) ou la  $k$  sous-diagonale (si  $k<0$ ).

```
>> A = [4 5 6 7 ; 3 4 5 6
```

```
2 3 4 5; 1 2 3 4]
```

```
A =
```

```
4 5 6 7
```

```
3 4 5 6
```

```
2 3 4 5
```

```
1 2 3 4
```

```
>> diag(A,1)
```

```
ans =
```

```
5
```

```
5
```

```
5
```

```
>> diag(A,-2)
```

```
ans =
```

2

2

On dispose également de la commande **tril** permet d'obtenir la partie triangulaire inférieure (l pour lower) d'une matrice. La commande **triu** permet d'obtenir la partie triangulaire supérieure (u pour upper) d'une matrice.

```
>> A = [ 2 1 1 ; -1 2 1 ; -1 -1 2 ]
```

```
A =
```

```
2 1 1  
-1 2 1  
-1 -1 2
```

```
>> triu(A)
```

```
ans =
```

```
2 1 1  
0 2 1  
0 0 2
```

```
>> tril(A)
```

```
ans =
```

```
2 0 0  
-1 2 0  
-1 -1 2
```

Comme pour la commande **diag**, les commandes **triu** et **tril** admettent un second paramètre  $k$ . On peut ainsi obtenir la partie triangulaire supérieure (ou inférieure) à partir de la  $k$  diagonale. Ainsi,

```
>> A = [ 2 1 1 ; -1 2 1 ; -1 -1 2 ]
```

```
A =
```

```
2 1 1  
-1 2 1  
-1 -1 2
```

```
>> tril(A,-1)
```

```
ans =
```

```
0 0 0  
-1 0 0  
-1 -1 0
```

```
>> tril(A,1)
```

```
ans =
```

```
0 1 1  
0 0 1  
0 0 0
```

On obtient la transposée de la matrice A à coefficients réels en tapant **A'**. Si la matrice est à coefficients complexes, **A'** retourne la matrice adjointe de A.

```
>> A = [0 1 2; -1 0 1; -2 -1 0]
```

```
A =
```

```
0 1 2  
-1 0 1  
-2 -1 0
```

```
>> A' %transposée de A
```

```
ans =
```

```
0 -1 -2  
1 0 -1  
2 1 0
```

### 3.6.4 Opérations et fonctions appliquées sur les matrices

Opération	Résultat	Condition
A + B	Matrice dont les éléments sont $a_{ij} + b_{ij}$	A et B sont de même taille
A - B	Matrice dont les éléments sont $a_{ij} - b_{ij}$	A et B sont de même taille
A + c ou A - c (c'est un nombre)	Matrice dont les éléments sont $a_{ij} + c$ ou $a_{ij} - c$	
A * B	Produit matricielle de A et B	Nombre de colonne de A=nombre de lignes de B
A * c = c * A	Matrice dont les éléments sont $a_{ij} * c$	

$A \cdot B$	Matrice dont les éléments sont $a_{ij} * b_{ij}$	A et B sont de même taille
$A^B$	<b>Matrice dont les éléments sont <math>(a_{ij})^{bij}</math></b>	<b>A et B sont de même taille</b>
$A^n$ ( $n > 0$ )	$A * A * \dots * A$ (n fois)	A est carrée
$A^n$ ( $n < 0$ )	$A^{-1} * A^{-1} * \dots * A^{-1}$ (n fois)	A est inversible
$B/A$	$B * A^{-1}$	A est inversible
$A B$	$A^{-1} * B$	A est inversible
$A/B$	Matrice dont les éléments sont $a_{ij} / b_{ij}$	A et B sont de même taille
$A B$	Matrice dont les éléments sont $b_{ij} / a_{ij}$	A et B sont de même taille

### Fonctions appliquées sur les matrices

- $\det(A)$  : déterminant de A,  
 $\text{rank}(A)$  : rang de A,  
 $\text{trace}(A)$  : somme des éléments de la diagonale de A,  
 $\text{inv}(A)$  : inverse de A,  
 $[V, D] = \text{eig}(A)$  : Valeurs propres et vecteurs propre de A , avec  $A = V*D*V^{-1}$  ,  
(D: matrice diagonale contenant les valeurs propres,  
V: matrice dont les colonnes sont les vecteurs propres)  
 $\text{Poly}(A)$  : Polynôme caractéristique de A.  
 $\text{expm}(A)$  : exponentielle matricielle de A.  
 $[m, n] = \text{size}(A)$  : la taille de la matrice A (m \*n)

## 4 Représentations graphiques Graphique 2D et 3D

### 4.1 Représentations graphiques Graphique 2D

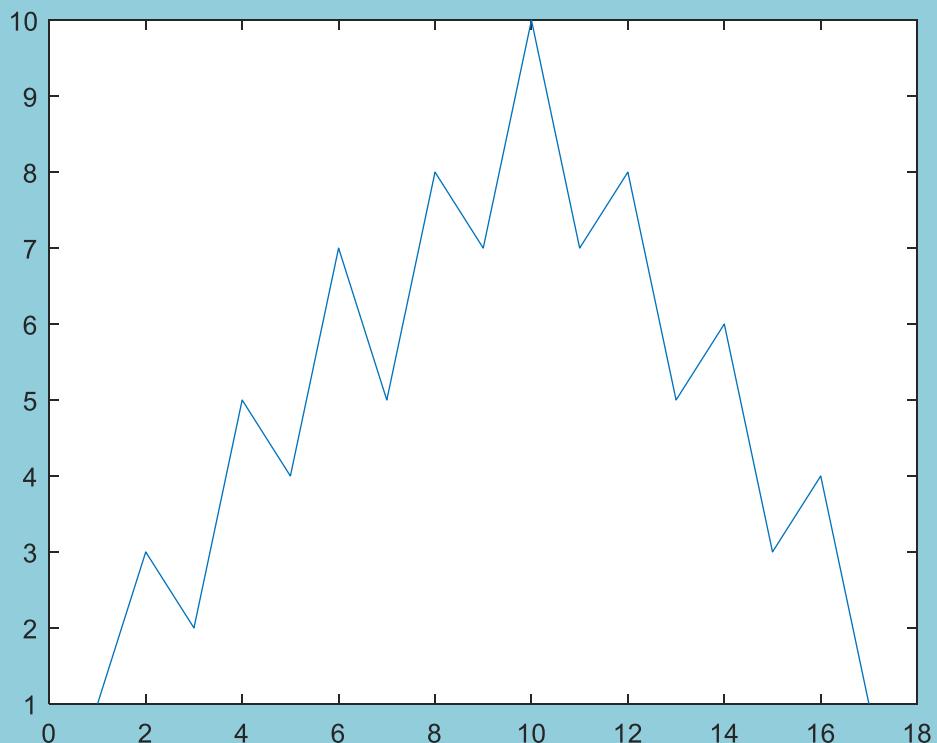
#### 4.1.1 La fonction plot

- ✓ Si  $y$  est un vecteur,  $\text{plot}(y)$  produit un graphique linéaire des éléments de  $y$  selon l'index des éléments de  $y$ .
- ✓ Si vous précisez deux vecteurs en tant qu'arguments,  $\text{plot}(x,y)$  produit un graphique de  $y$  selon  $x$ .

#### Exemple 1

```
>>y= [1 3 2 5 4 7 5 8 7 10 7 8 5 6 3 4 1];  
>> plot(y)
```

La figure est donnée comme suit

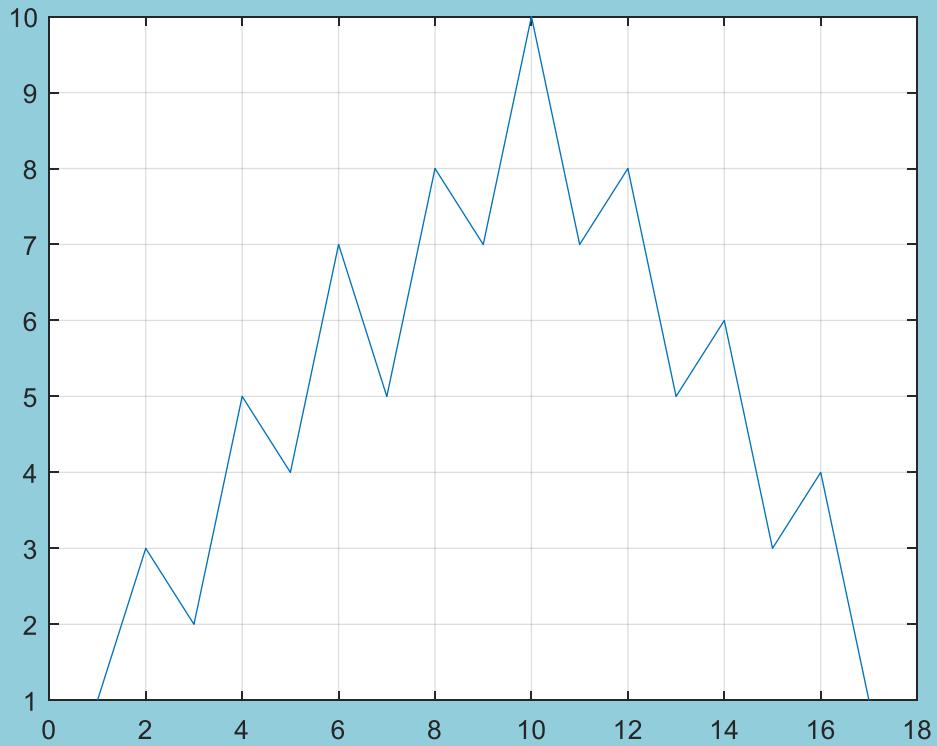


On ajoute la commande

```
grid on
```

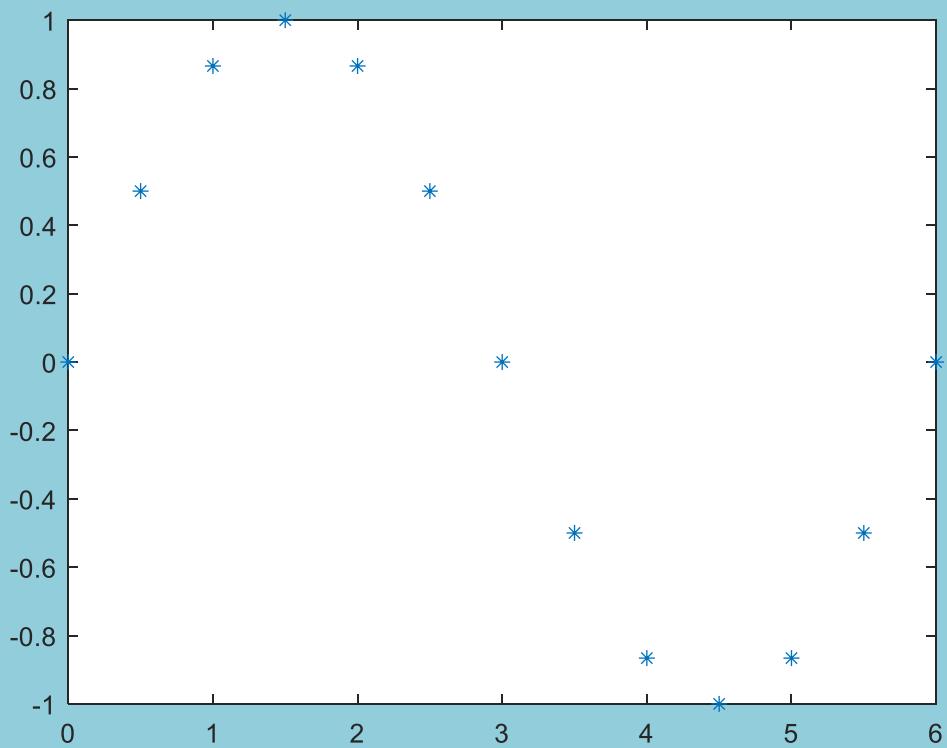
```
>>y=[1 3 2 5 4 7 5 8 7 10 7 8 5 6 3 4 1];
```

```
>> plot(y); grid on;
```



## Exemple 2

```
>> t=0:0.5:6;  
>> w=pi/3;  
>> y=sin(w*t);  
>> plot(t,y,'*')
```



Traçons la fonction sinus dans l'intervalle  $[-\pi, \pi]$  avec un pas de 0,01

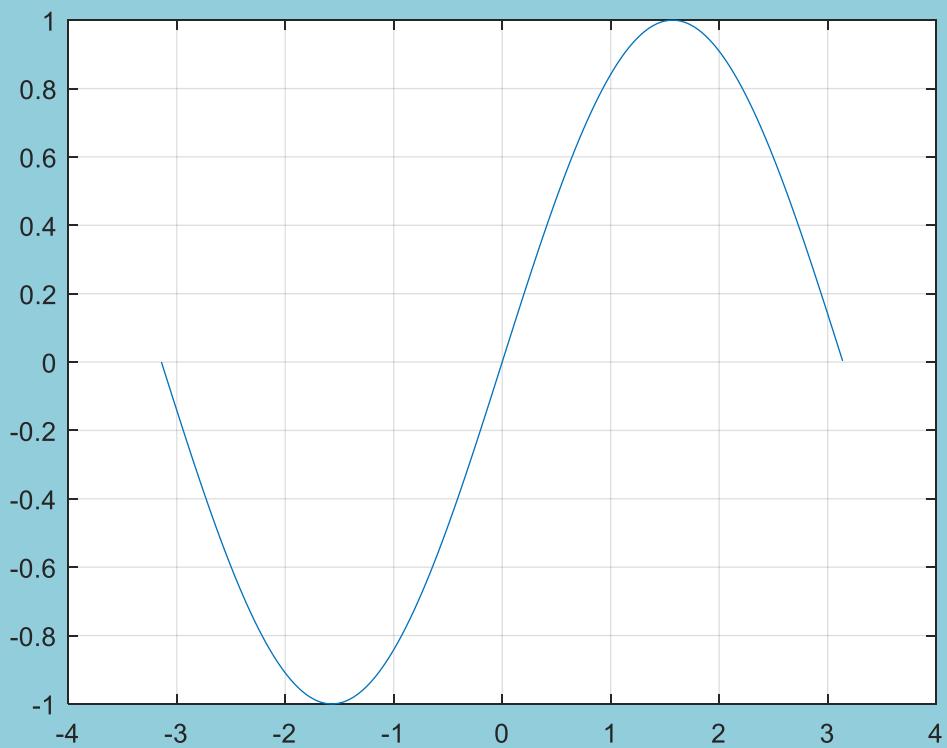
```
>>x=-pi : 0.01 : pi;
```

```
>> y=sin(x);
```

```
>>plot(x,y)
```

```
>>grid on ;
```

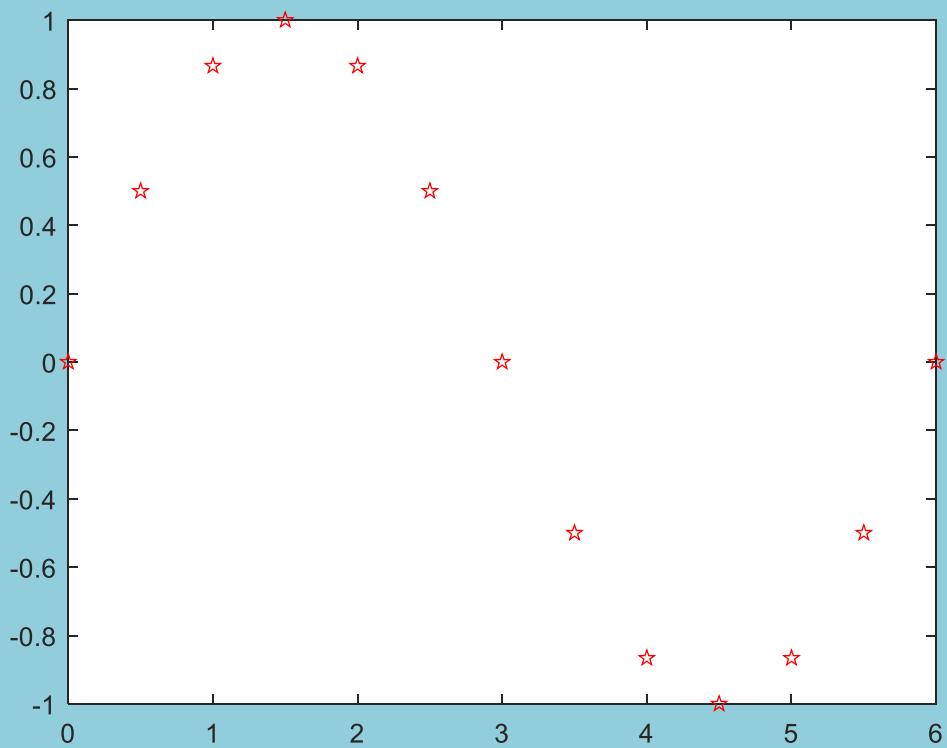
Le pas étant faible, la courbe est parfaitement tracée.



### Exemple 3

```
>> t=0:0.5:6;  
>> w=pi/3;  
>> y=sin(w*t);  
>>plot(t,y, 'rp')
```

le troisième argument permet de spécifier la couleur du tracé et le symbole de la représentation.



```
>> plot(t,Y,S)
```

**Différents types de lignes, symboles de tracé et couleurs peuvent être obtenus avec `plot(t,Y,S)` où S est une chaîne de caractères composée d'un élément à partir de l'une ou de toutes les 3 colonnes suivantes :**

<b>b</b>	<b>blue</b>	.	<b>point</b>	-	<b>solid</b>
<b>g</b>	<b>green</b>	o	<b>circle</b>	:	<b>dotted</b>
<b>r</b>	<b>red</b>	x	<b>x-mark</b>	-.	<b>dashdot</b>
<b>c</b>	<b>cyan</b>	+	<b>plus</b>	--	<b>dashed</b>
<b>m</b>	<b>magenta</b>	*	<b>star</b>	(none)	<b>no line</b>
<b>y</b>	<b>yellow</b>	s	<b>square</b>		
<b>k</b>	<b>black</b>	d	<b>diamond</b>		

w	<b>white</b>	v	<b>triangle (down)</b>
^		^	<b>triangle (up)</b>
<		<	<b>triangle (left)</b>
>		>	<b>triangle (right)</b>
p		p	<b>pentagram</b>
h		h	<b>hexagram</b>

#### 4.1.2 Superposition de plusieurs graphiques

A chaque nouvelle commande **plot**, la figure est remplacée par la dernière figure. Pour garder plusieurs courbes on utilise la commande **hold on**. Les plot suivants se superposent jusqu'à la désactivation par la commande **hold off** ou par la fermeture de la fenêtre.

```
>> x=linspace(-20,20,1000);

>> y1=x.*sin(x);

>> plot(x,y1,'b')

>> hold on;

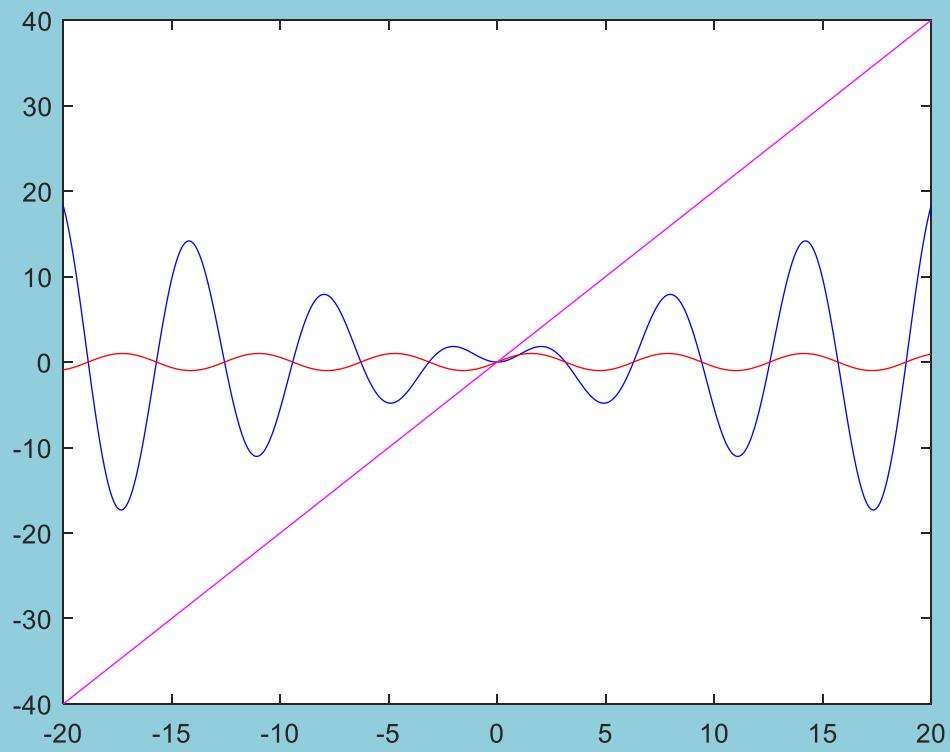
>> y2=sin(x);

>> plot(x,y2,'r');

>> hold on;

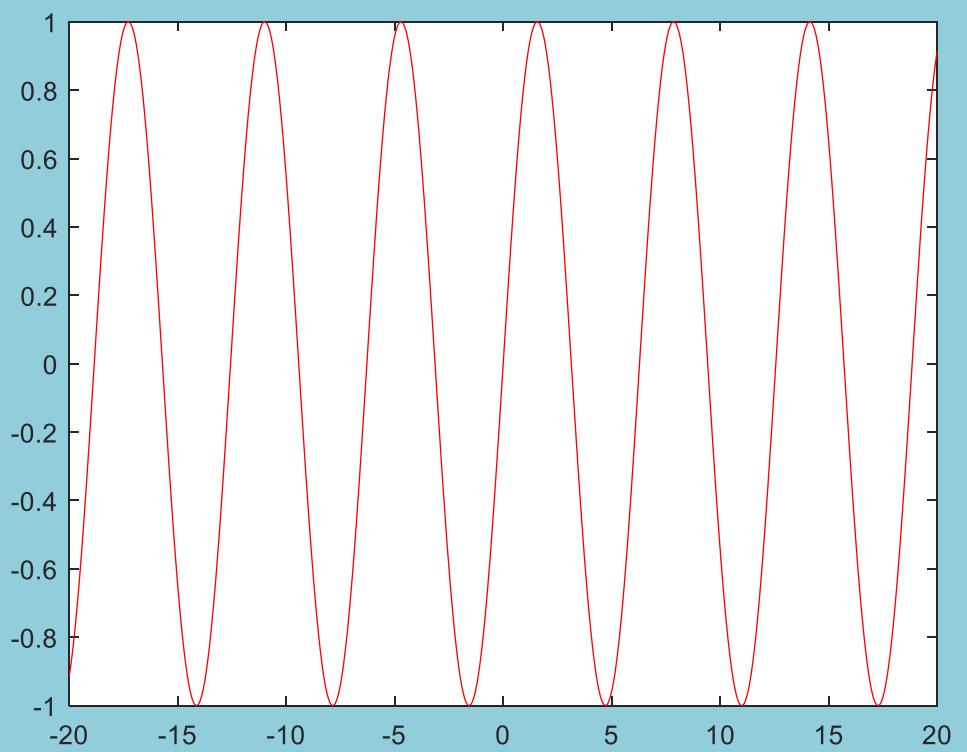
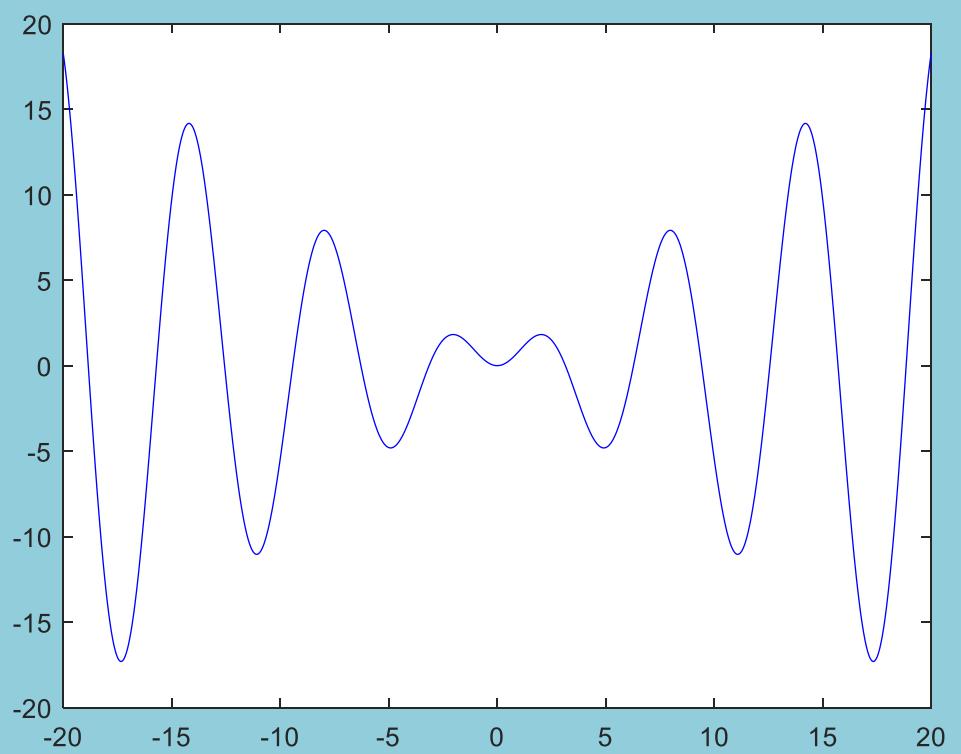
>> y3=2*x;

>> plot(x,y3,'y');
```



On peut également tracer ces courbes dans des figures différentes

```
>> x=linspace(-20,20,1000);
>> y1=x.*sin(x);
>>figure(1)
>> plot(x,y1,'b')
>> y2=sin(x);
>>figure(2)
>> plot(x,y2,'r');
```



### 4.1.3 Mise en forme de graphiques

L'insertion de labels, légende le dimensionnement des axes, peut être éditée de deux manière.

- La méthode la plus simple utilise directement les menus de l'interface de la figure(Edit et Insert).
- Toutes ces manipulations sont également réalisables à partir de la Command Window en ligne de commande.

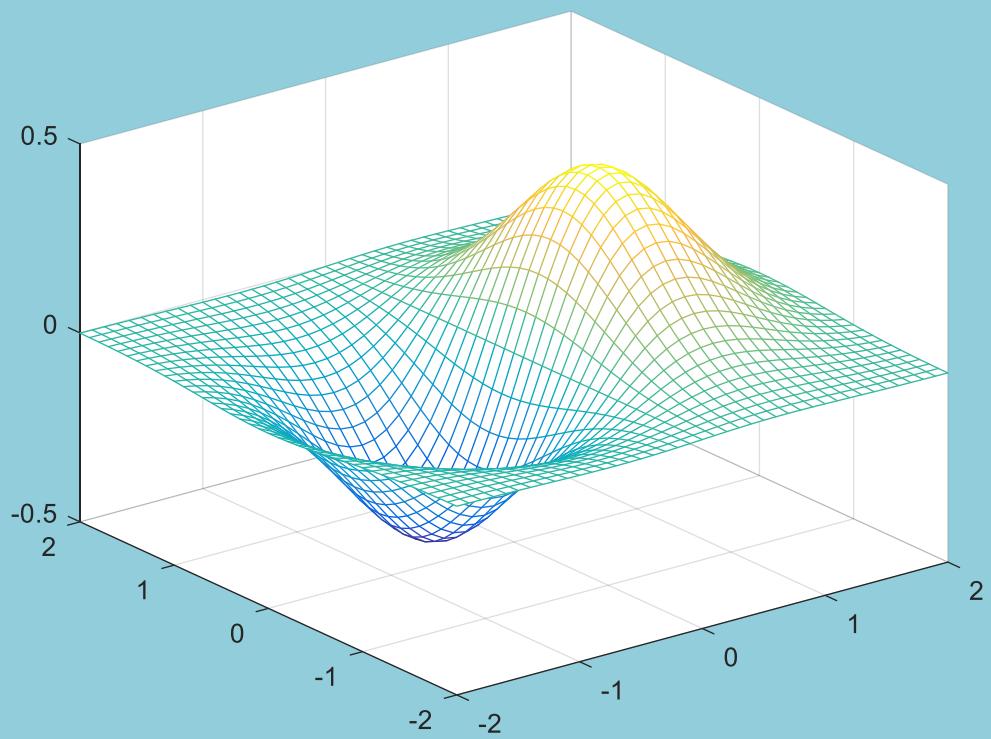
```
>>xlabel('valeur x');  
>>ylabel('valeur y');  
>>title(' mon graphique');  
>>legend('ma courbe');  
>>axis([xmin xmax ymin ymax])
```

## 4.2 Représentations graphiques graphique 3D

### Exemples

1)

```
>>x=[-2 : 0.1 :2];  
>>y=[-2 : 0.1 : 2];  
>>[X,Y]=meshgrid(x,y);  
>>Z=X.*exp(-X.^2-Y.^2) ;  
>>mesh(X,Y,Z)  
% Pour changer la couleur colormap([0.5 0.5 0.5]);
```



2)

```
>>x=[-2 : 0.1 :2];  
>>y=[-2 : 0.1 : 2];  
>>[X,Y]=meshgrid(x,y);  
>>Z=X.*exp(-X.^2-Y.^2)  
>>surf(X,Y,Z);
```

